Parserite kombinaatorid

Süntaksanalüüs

- Parseri ülesandeks on:
  - kontrollida kas sisendstring on süntaktiliselt korrektne;
  - konstrueerida sisendstringile vastav AST.
- Parserite soovitavad omadused:
  - BNF lähedane esitus;
  - parserite konstrueerimine olemasolevatest parseritest;
  - mittedetermineeritud grammatikate kasutamine;
  - kontekstist sõltuvate keelte äratundmine.

Parserite kombinaatorid

Parserite tüüp

$$\textbf{type}\ Parser\ a = String \to [(a, String)]$$

Triviaalsed parserid

$$\begin{aligned}
failp &:: Parser\ a \\
failp &= const\ [\,] \\
succp &:: a \to Parser\ a \\
succp\ x &= \lambda cs \to [(x, cs)] \\
epsilon &:: Parser\ () \\
epsilon &= succp\ ()
\end{aligned}$$

Parserite kombinaatorid

Elementaarparserid

$$
\begin{aligned}
&satp \;\; :: \;\; (Char \to Bool) \to Parser\ Char \\
&satp\ p\ [\,] \;\;\;\;\;\;\;\; = [\,] \\
&satp\ p\ (c : cs) = [(c, cs) \mid p\ c] \\
&symp \;\; :: \;\; Char \to Parser\ Char \\
&symp\ c = satp\ (\equiv c) \\
&tokp \;\; :: \;\; String \to Parser\ String \\
&tokp\ s \;\; = \lambda cs \to [(s, drop\ n\ cs) \mid s \equiv take\ n\ cs] \\
&\qquad\quad \mathbf{where}\ n = length\ s
\end{aligned}
$$

Parserite komponeerimine

Järjestikkompositsioon

```
infixr 6 <*>
(<*>) :: Parser a → Parser b → Parser (a, b)
(p1 <*> p2) cs = [((v1, v2), cs2) | (v1, cs1) ← p1 cs,
                                    (v2, cs2) ← p2 cs1]
```

Paralleelne kompositsioon

```
infixr 4 <|>
(<|>) :: Parser a → Parser a → Parser a
(p1 <|> p2) cs = p1 cs ++ p2 cs
```

Parserite transformaatorid

Väärtustega manipuleerimine

$$\textbf{infixr}\ 5\ <@$$
$$(<@) :: Parser\ a \to (a \to b) \to Parser\ b$$
$$(p\ <@\ f)\ cs = [(f\ v, cs') \mid (v, cs') \leftarrow p\ cs]$$

Näide

$$ac\_bc :: Parser\ (Char, Char)$$
$$ac\_bc = (symp\ \text{'a'} <|> symp\ \text{'b'}) <*> symp\ \text{'c'}$$

Parserite transformaatorid

Näide

```
data Tree = Nil | Bin Tree Tree
parens :: Parser Tree
parens = (       symp '('
             <*> parens
             <*> symp ')'
             <*> parens
         )                  <@ (λ(_, (x, (_, y))) → Bin x y)
          <|>   epsilon <@ const Nil
```

Parserite komponeerimine

Järjestikkompositsiooni lühendid

```
infixr 6 <*
(<*)    :: Parser a → Parser b → Parser a
p <* q  = p <*> q <@ fst
infixr 6 *>
(*>)    :: Parser a → Parser b → Parser b
p *> q  = p <*> q <@ snd
infixr 6 <:*>
(<:*>)  :: Parser a → Parser [a] → Parser [a]
p <:*> q = p <*> q <@ uncurry (:)
```

Parserite transformaatorid

Iteratsioon

$$
\begin{aligned}
many &:: Parser\ a \to Parser\ [a] \\
many\ p &= many1\ p <|> succp\ [] \\
many1 &:: Parser\ a \to Parser\ [a] \\
many1\ p &= p <:*> many\ p
\end{aligned}
$$

Üldistatud järjestik- ja paralleelkompositsioon

$$
\begin{aligned}
seqp &:: [Parser\ a] \to Parser\ [a] \\
seqp &= foldr\ (<:*>)\ (succp\ [\,]) \\
altp &:: [Parser\ a] \to Parser\ a \\
altp &= foldr\ (<|>)\ failp
\end{aligned}
$$

Lihtsaid parsereid

Võtmesõnad

$$tokp :: String \to Parser\ String$$
$$tokp = seqp \circ map\ symp$$

Identifikaatorid

$$ident :: Parser\ String$$
$$ident = satp\ isAlpha <:*> many\ (satp\ isAlphaNum)$$

Lihtsaid parsereid

Naturaalarvud

$$
\begin{aligned}
&digit &&:: Parser\ Int\\
&digit &&= satp\ isDigit\ <@\ \lambda x \rightarrow ord\ x - ord\ \texttt{'0'}\\
&natural &&:: Parser\ Int\\
&natural &&= many1\ digit\ <@\ foldl1\ (\lambda a\ b \rightarrow 10 * a + b)
\end{aligned}
$$

Täisarvud

$$
\begin{aligned}
&sign &&:: Parser\ (Int \rightarrow Int)\\
&sign &&= symp\ \texttt{'-'}\ <@\ const\ negate\ <|>\ succp\ id\\
&integer &&:: Parser\ Int\\
&integer &&= sign\ <*>\ natural\ <@\ uncurry\ (\$)
\end{aligned}
$$

Lihtsaid parsereid

Reaalarvud

```
fract :: Parser Float
fract = many1 digit <@ foldr op 0
      where d `op` x = (x + fromIntegral d) / 10
float :: Parser Float
float = (integer <@ fromIntegral)
            <*> (symp '.' *> fract <|> succp 0)
            <@ uncurry (+)
```

Lihtsaid parsereid

Tühisümbolid

$$
\begin{aligned}
sp &:: Parser\ a \to Parser\ a \\
sp &= (many\ (satp\ isSpace)*>) \\
spsym &:: Char \to Parser\ Char \\
spsym &= sp \circ symp \\
sptok &:: String \to Parser\ String \\
sptok &= sp \circ tokp \\
spident &:: Parser\ String \\
spident &= sp\ ident \\
spint &:: Parser\ Int \\
spint &= sp\ integer
\end{aligned}
$$

Parserite transformaatorid

Optsioon ja sulud

```
optp   :: Parser a → Parser [a]
optp p =      p <@ (:[])
          <|> succp []
pack   :: Parser a → Parser b → Parser c → Parser b
pack s1 p s2 = s1 *> p <* s2
```

Näide

```
paren p = pack (spsym '(') p (spsym ')')
brack p = pack (spsym '[') p (spsym ']')
block p = pack (sptok "begin") p (sptok "end")
```

Parserite transformaatorid

Eraldajaga jadad

$$
\begin{aligned}
&listp :: Parser\ a \to Parser\ b \to Parser\ [a] \\
&listp\ p\ s = \quad p <:*> many\ (s *> p) \\
&\qquad\qquad <|> succeed\ [\,]
\end{aligned}
$$

Näide

$$
\begin{aligned}
commaList\ p &= listp\ p\ (symp\ ',') \\
semicList\ p &= listp\ p\ (symp\ ';') \\
listExpr &= brack \circ commaList \\
pasBlock &= block \circ semicList
\end{aligned}
$$

Aritmeetilised avaldised

Grammatika

$$
\begin{array}{rcl}
\text{expr} & = & \text{int} \\
& | & \text{expr} + \text{expr} \\
& | & \text{expr} - \text{expr} \\
& | & \text{expr} * \text{expr} \\
& | & \text{expr} \ / \ \text{expr} \\
& | & (\text{expr})
\end{array}
$$

Abstraktne süntaksipuu

$$
\begin{array}{rcl}
\textbf{data}\ \mathit{Expr} & = & \mathit{Con}\ \mathit{Int} \\
& | & \mathit{Expr} : + : \mathit{Expr} \\
& | & \mathit{Expr} : - : \mathit{Expr} \\
& | & \mathit{Expr} : * : \mathit{Expr} \\
& | & \mathit{Expr} : / : \mathit{Expr}
\end{array}
$$

Aritmeetilised avaldised

Parser ver. 1

$$
\begin{aligned}
expr &= atom \mathbin{<*>} oper \mathbin{<*>} expr \\
&\qquad \mathbin{<@} (\lambda(a,(op,e)) \to op\ a\ e) \\
&\quad \mathbin{<|>} atom \\
oper &= spsym\ \text{'+'} \mathbin{<@} const\ (:+:) \\
&\quad \mathbin{<|>} spsym\ \text{'-'} \mathbin{<@} const\ (:-:) \\
&\quad \mathbin{<|>} spsym\ \text{'*'} \mathbin{<@} const\ (:*:) \\
&\quad \mathbin{<|>} spsym\ \text{'/'} \mathbin{<@} const\ (:/:) \\
atom &= spint \mathbin{<@} Con \\
&\quad \mathbin{<|>} paren\ expr
\end{aligned}
$$

Parserite transformaatorid

Eraldajaga jadad

```
chainl :: Parser a → Parser (a → a → a) → Parser a
chainl p s  = p <*> many (s <*> p)
                    <@ uncurry (foldl (flip ap2))
              where ap2 (op, y) = (`op` y)
chainr :: Parser a → Parser (a → a → a) → Parser a
chainr p s = many (p <*> s) <*> p
                    <@ uncurry (flip (foldr ap1))
              where ap1 (x, op) = (x `op`)
```

Aritmeetilised avaldised

Parser ver. 2

```
expr :: Parser Expr
expr = chainl term (     spsym '+' <@ const (:+:)
                     <|> spsym '-' <@ const (:-:))
term :: Parser Expr
term = chainl fact (     spsym '*' <@ const (:*:)
                     <|> spsym '/' <@ const (:/:))
fact :: Parser Expr
fact = spint <@ Con <|> paren expr
```

Aritmeetilised avaldised

Aritmeetiliste avaldiste väärtustaja

```
expr :: Parser Int
expr = chainl term (   spsym '+' <@ const (+)
                   <|> spsym '-' <@ const (-))
term :: Parser Int
term = chainl fact (   spsym '*' <@ const (*)
                   <|> spsym '/' <@ const div)
fact :: Parser Int
fact = spint <@ id <|> paren expr
```

Parserite efektiivsus

Efektiivsust parandavaid kombinaatoreid

$$
\begin{aligned}
&\mathit{just} && :: \ \mathit{Parser}\ a \to \mathit{Parser}\ a \\
&\mathit{just}\ p && = \mathit{filter}\ (\mathit{null} \circ \mathit{snd}) \circ p \\
&\mathit{first} && :: \ \mathit{Parser}\ a \to \mathit{Parser}\ a \\
&\mathit{first}\ p\ cs && = [\mathit{head}\ r \mid \neg\ (\mathit{null}\ r)] \\
& && \mathbf{where}\ r = p\ cs \\
&\mathit{greedy} && = \mathit{first} \circ \mathit{many} \\
&\mathit{greedy1} && = \mathit{first} \circ \mathit{many1} \\
&\mathit{compulsion} && = \mathit{first} \circ \mathit{option} \\
&\mathit{sp} && = (\mathit{greedy}\ (\mathit{satp}\ \mathit{isSpace}) *>)
\end{aligned}
$$